Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

BF

CA 2303149 C 2003/10/21

(11)(21) 2 303 149

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

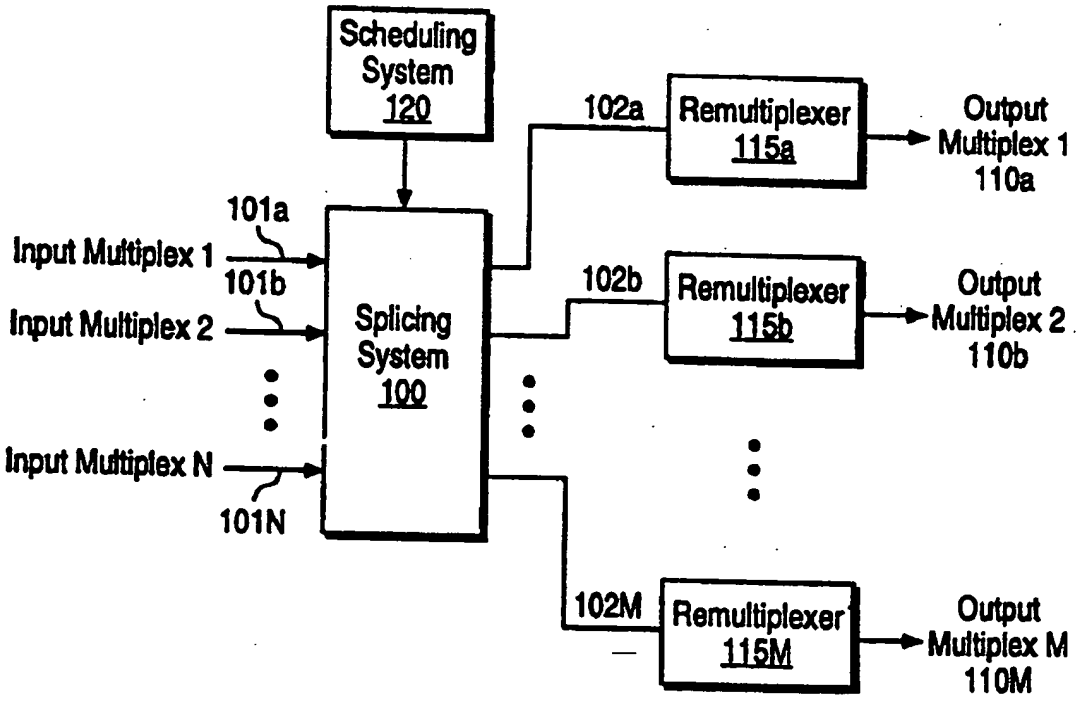
(86) Date de dépôt PCT/PCT Filing Date: 1998/08/27
(87) Date publication PCT/PCT Publication Date: 1999/03/25
(45) Date de délivrance/Issue Date: 2003/10/21
(85) Entrée phase nationale/National Entry: 2000/03/09
(86) N° demande PCT/PCT Application No.: US 1998/017757
(87) N° publication PCT/PCT Publication No.: 1999/014955
(30) Priorité/Priority: 1997/09/12 (08/928,414) US

(51) Cl.Int.^{7}/Int.Cl.^{7} H04N 7/58
(72) Inventeurs/Inventors:
KRAUSE, EDWARD A., US;
MONTA, PETER A., US

(73) Propriétaire/Owner:
IMEDIA CORPORATION, US

(74) Agent: MACRAE & CO.

(54) Titre : COLLAGE SANS SOLUTION DE CONTINUITE DE PROGRAMMES VIDEO COMPRIMES
(54) Title: SEAMLESS SPLICING OF COMPRESSED VIDEO PROGRAMS



(57) Abrégé/Abstract:
A system for seamless splicing of compressed video programming. Various parameters in the splice data are adjusted for proper handling in accordance with the splice. Permissible splice points in both a current and a next program are determined which are most consistent with a desired splice time in order to seamlessly effect the transition from the current to the next program.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04N 7/58 — A1

(11) International Publication Number: WO 99/14955

(43) International Publication Date: 25 March 1999 (25.03.99)

(21) International Application Number: PCT/US98/17757

(22) International Filing Date: 27 August 1998 (27.08.98)

(30) Priority Data:
08/928,414 12 September 1997 (12.09.97) US

(71) Applicant: IMEDIA CORPORATION [US/US]; 188 Embarcadero, San Francisco, CA 94107 (US).

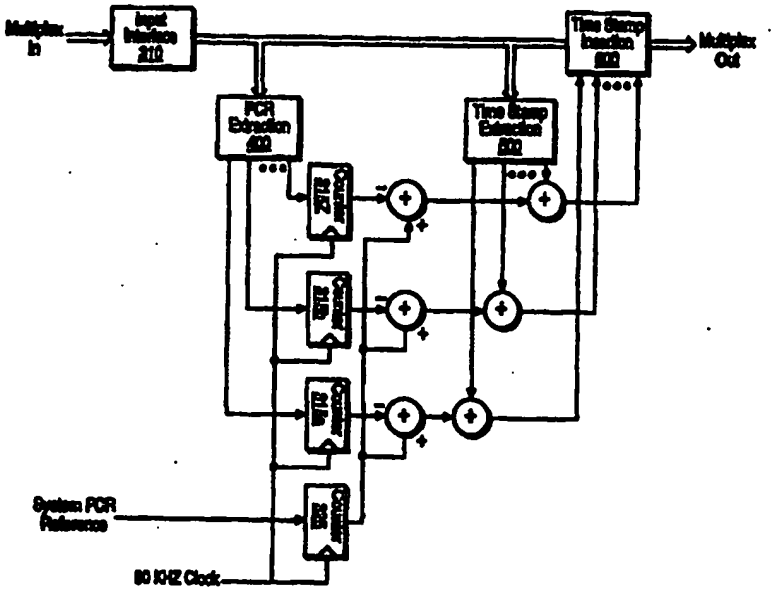
(72) Inventors: KRAUSE, Edward, A.; 35 Burgoyne Court, San Mateo, CA 94402 (US). MONTA, Peter, A.; Apartment 315, 1550 Bay Street, San Francisco, CA 94123 (US).

(74) Agent: WAWRZYNIAK, Richard, E.; McCutchen, Doyle, Brown & Enersen, LLP, Three Embarcadero Center, San Francisco, CA 94111 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: SEAMLESS SPLICING OF COMPRESSED VIDEO PROGRAMS



(57) Abstract

A system for seamless splicing of compressed video programming. Various parameters in the splice data are adjusted for proper handling in accordance with the splice. Permissible splice points in both a current and a next program are determined which are most consistent with a desired splice time in order to seamlessly effect the transition from the current to the next program.

SEAMLESS SPLICING OF COMPRESSED VIDEO PROGRAMS

BACKGROUND ART

1. Technical Field

The present invention relates to the distribution of video programming. More particularly, the present invention relates to the seamless splicing of compressed digital video programming.

2. Background

Splicing systems are used in cable television head ends, satellite uplink facilities, television stations, and editing laboratories. They are used for switching between two different programs, inserting advertisements into a program, or simply editing program content. Today, almost all program splicing is applied to uncompressed analog or digital sources. In the future, there will be a need to splice different programs that exist in compressed form.

A number of problems are presented by efforts to splice compressed programs. For example, programs that are digitally compressed according to standards such as MPEG contain many codeword sequences that are only meaningful to a decoder when interpreted in combination with previous and following information in the compressed bit stream. In particular, a compression standard may specify that only the changes that have occurred since a prededing time instant be encoded and transmitted. For this reason, the current program cannot simply be interrupted at arbitrary locations, and similarly, the new

program cannot be simply inserted at the point of interruption. Otherwise, visible and audible artifacts are likely to be noticed during presentation of the reconstructed program.

A second problem arises because compressed programs use rate buffers at both the sending and receiving stations in order to deal with short term fluctuations in the compression ratio. Therefore, when a compressed program is examined at any given time, it is difficult to determine the intended presentation time of a particular data unit without accounting for the time required to propagate through the rate buffer at the receiver. Without this information, the time of the splice, and therefore the precise point at which the current program should be interrupted, cannot be accurately determined.

During the data compression step, the encoder is responsible for insuring that the rate buffer at a receiver remains within acceptable occupancy limits to avoid both overflow and underflow. Generally, compression standards are designed such that all receivers receiving the same compressed bit stream exhibit identical buffer occupancy levels. This introduces a third problem, since the programs to be spliced are generally encoded independently. It therefore cannot be assumed that neither overflow nor underflow will be triggered by the transition between the two programs. This is because the encoder utilized for the current program assumes a particular buffer occupancy level at the precise time of the splice, while the encoder utilized for the next program may well assume another. Unless the two assumed occupancy levels are identical, overflow or underflow becomes possible, which may result in visible and/or audible artifacts.

A fourth complexity is introduced in connection with splicing compressed programming if a remultiplexer or stream reformatting system is coupled to the output of the

splicing system. A remultiplexer is used to recombine several programs into a common signal while a reformatting system may be used to interface a program to a particular distribution system. Either of these systems may cease to function correctly when simultaneously or sequentially provided with two or more compressed programs that have been synchronized to different timing references. For example, if one or more programs were encoded based on a display rate of 29.97001 frames per second while one or more different programs were encoded based on a display rate of 29.96999 frames per second, then a problem may arise even though this amount of variation may be well within the capture range of the intended receivers. If the remultiplexer retransmits the sequences at any rate other than the original rate assumed by each of the respective encoders, then the data rate mismatch occurring at the remultiplexer will eventually overflow or underflow any buffering capacity that exists. Since even a simple splicing system must perform certain reformatting operations, and since many splicing systems are likely to be integrated into larger switches and remultiplexers, a solution to this problem is needed to avoid limiting the usefulness and versatility of the splicer. Furthermore, methods for dealing with the many problems of splicing can also help to solve the problems of reformatting or remultiplexing.

Because of the many problems associated with the splicing of compressed digital programs, non-seamless solutions have been proposed. For example, the duration of the splice can be extended by inserting a brief silent black interval between the current program and the next program. During this interval, the receiver rate buffer is allowed to empty so that the new program can begin with a known buffer occupancy level. This interval can also simplify the task of creating independence among the two encoded bit streams. However, even the implementation of this non-seamless splicing technique does not

solve the problems of splice time inaccuracy or mismatched timing references.

It would be advantageous, and is therefore an object of the present invention, to provide a fully seamless splicing system that overcomes the above-noted problems.

DISCLOSURE OF THE INVENTION

From the foregoing it can be seen that it would be desirable and advantageous to develop a method and system to provide fully seamless splicing for compressed video programming. The present invention is thus directed toward a compressed video program splicing system in which a second or "next" compressed program is spliced in to follow a first or "current" compressed program without the introduction of video or audio artifacts caused by the several phenomena discussed in the previous section.

The scheduled splice time in general will not coincide with the permissible splice times in the current and next programs. Two of these times must be shifted to achieve triple coincidence, which is required for seamless splicing. One way to do this is to search for a pair of splice times in the two programs that are both relatively close together and relatively close to the scheduled splice time. Then the actual splice time can be shifted from the scheduled splice time to coincide with the permissible splice time in the current program, and the next program can be delayed as needed to achieve the desired triple coincidence. However, nearly as good results can be achieved with a simpler method and apparatus, so that in a preferred embodiment of the present invention, only the relative position of the scheduled splice time and the permissible splice points of the current program are used.

One implementation of the seamless-splicing system of the present invention utilizes two program processors that are responsive to a scheduler for determining when the current program is to be cut out and the next program is to be cut in based on a scheduled splice time. At any given time one of the program processors is in an active mode, processing the current program, while the other program processor is in a standby mode,

waiting for the splice command from the active processor. The active processor ensures that all of the data required to reconstruct all of the frames in the spliced program is included in the output data stream. The active program processor is also responsible for issuing a splice command signal to the standby program processor at the appropriate time.

The standby program processor ensures that the next data stream will contain no references to prior data that are not included after the splice point. The standby program processor also determines where in the next program the appropriate point is for appending the next program to the end of the current program. The standby program processor then carries out various functions such as shifting the next program stream in time and adjusting the next-program content while ensuring that all embedded parameters in the compressed program stream are properly adjusted in order to reflect the changes that have occurred.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the present invention will be apparent from the following detailed description, in which:

Figure 1 is a general purpose multi-program splicing system which may incorporate the present invention.

Figure 2 is a more detailed block diagram of a splicing system which may incorporate the present invention.

Figure 3 is a block diagram of components used for normalizing multiple time stamps to a single common reference in accordance with one aspect of the present invention.

Figure 4 is a flow diagram for the extraction of program clock references from a received data stream.

Figure 5 is a flow diagram for the process of extracting time stamps from a received data stream.

Figure 6 is a flow diagram for the process of time stamp insertion in accordance with one aspect of the present invention.

Figure 7 is a more detailed block diagram of an output module for use in the splicing system of the present invention.

Figure 8 shows several exemplary frame sequences to demonstrate some of the details of splicing compressed program data.

Figure 9 is a block diagram of a program splicer for use in accordance with the present invention.

Figure 10 is a block diagram of the components of a program processor for use in accordance with the present invention.

Figure 11 is a timing diagram of the signals corresponding to the operation of the circuit of Figure 10 in accordance with the present invention.

Figure 12 is a flow diagram for adjusting a temporal reference parameter after a program splice has occurred in accordance with the splicing system of the present invention.

Figure 13 is a flow diagram for time stamp adjustment following a program splice in accordance with the splicing system of the present invention.

Figure 14 is a flow diagram for a procedure to statistically multiplex data packets corresponding to different programs.

NOTATION AND NOMENCLATURE

The detailed descriptions which follow are presented largely in technical terms relating to the handling of compressed video programming. To aid in the understanding of this description, the following definitions are provided:

Seamless splicing. Producing a program by cutting out a current program and cutting in a next program on a field boundary without the production of visible or audible artifacts.

Permissible splice point. (PSP) A field boundary in a program at which the compressed program can be cut in or cut out in such a way that the compressed spliced program contains all the data needed for reconstruction of all of the fields in the spliced program.

Statistical multiplexing. The process by which a multiplicity of compressed programs can be combined into one signal, allotting more channel capacity to more complex programs and less capacity to less complex programs.

Field, Frame. A frame is a single complete TV image. When interlace is used, each field comprises either the even lines or the odd lines of a frame.

Field/frame mode: An adaptive process used by some video compression systems when encoding interlaced television signals. Either field-based or frame-based processing techniques are selected, usually on a block-by-block or frame-by-frame basis.

MPEG. Motion Picture Experts Group. The compression system adopted by the International Standardization Organization for digital television signals.

Data stream. A sequence of data symbols, either in compressed or uncompressed form, representing a single channel of video, audio, or other type of data information.

Program. The collection of data streams containing all video, audio, and other information associated with the presentation of a television production.

Compressed program. The data streams representing a program in which the data rate has been reduced without serious loss of image and sound quality.

Current program. The program transmitted just before the splice.

Next program. The program transmitted just after the splice.

Rate buffer. A buffer used to mediate between different transmission rates.

Buffer overflow, underflow. The situation that arises when a buffer is too full to accept additional data that arrives or is empty at a time when additional data is needed.

Remultiplexer. A multiplexer used to combine several concurrent programs into one signal.

Reformatter. Apparatus that reconfigures a signal to accommodate the requirements of a particular distribution system.

Multiplexing. In general, combining several data streams into one.

Mux, demux. In general, performing and undoing the act of multiplexing.

Input, Output Multiplex. Ungrammatical terms used in MPEG to denote the input and output programs, each consisting of video data, audio data, and other data required to implement the decoding process. The distinction between "multiplex" used in this sense and "multiplex" used to mean the process of combining will be clear from the context.

Packet. A block of data into which a portion of the information to be transmitted is grouped. Each packet contains a header and a payload. The header identifies the packet completely so that original data streams may be correctly reassembled from the payloads of packets. Several different data streams can be transmitted in one stream of packets.

Scheduling system. The means by which the intended splicing time is generated. In many cases, the intended splicing time is produced by a human editor. In other cases, the intended splicing time is derived from information embedded in one or more data streams included in a program transmission. The list of splicing times for a spliced program is sometimes called a playlist. The scheduling system is not a part of the invention.

Presentation time. The time at which decoded images are displayed at the receiver.

Decoding time. The time at which compressed images are decoded at the receiver. Images are always decoded in

the same order that they are received. Decoding time is different from presentation time when the images are not received in presentation order.

Time stamp. Ancillary data embedded in the data streams of a program to indicate the intended presentation time or decoding time of a particular element of the program.

DTS, PTS. Decoding and presentation time stamps. May be included with program data.

Clock reference. Data representing the absolute clock time at the instant when particular program data is received.

PCR. Program clock reference. The clock reference applying to a particular program. A receiver which decodes or presents this program must adjust its own internal timing reference to match the received PCR's.

Intra Frame (I-Frame). A frame that is coded using only data of a single original frame and that therefore can be decoded independently of any other data in the program.

Predicted Frame (P-Frame). A frame that is coded using data from the corresponding original frame and one or more preceding frames.

Bi-directionally Predicted Frame (B-Frame). Like a P-frame, but using data from both preceding and following frames.

Anchor Frame. An I- or P-frame that is used to predict another frame. B-frames have two anchor frames. These are the closest preceding I- or P-frame and the closest following I- or P-frame.

PAT. Program association table. A table defined by the MPEG standard to identify the programs included in a received multiplex.

PMT. Program map table. A table defined by the MPEG standard to identify the data stream components of a particular program included in a received multiplex.

PID. Packet identification data. A unique MPEG data code included in the header of each packet of a particular data stream. Most PID's associated with the data streams of a particular program can be inferred by decoding the PAT and PMT tables of the same multiplex. All packets corresponding to the PAT data stream are assigned a PID value of

0. The PID's assigned to the PMT data streams of each program can be inferred from the PAT.

Splice command. A pulse that triggers the actual splice.

Sequence end code. A data code embedded in a compressed MPEG data stream to indicate the end of a video segment.

FIFO. Literally first in first out. A buffer used to implement a variable delay in a data stream.

GOP. Group of pictures. In MPEG, a sequence of coded fields beginning with an I-frame. A closed GOP can be reconstructed without referencing any data not contained within the GOP. An open GOP may include one or more B-frames to be displayed before the first I-frame received at the start of the GOP. Such B-frames cannot be reconstructed without referencing the last anchor frame of the preceding GOP.

Temporal Reference. An MPEG codeword included in the header of each coded video image to indicate the order of display.

VBR. Variable bit rate. A characteristic of a compression system in which the coded data rate varies with image complexity.

MODES FOR CARRYING OUT THE INVENTION

Methods and apparatus are disclosed for implementing seamless splicing of compressed video programming. Although the present invention is described predominantly in terms of an embodiment for splicing MPEG compressed digital video programs, the concepts and methods of the present invention are suitable for use in seamless splicing systems which rely on other compression technologies. Throughout this Detailed Description, numerous specific details are set forth such as particular data types and various pieces of control information, in order to provide a thorough understanding of the present invention. To one skilled in the art, however, it will be understood that the present invention may be practiced without such specific details. In other instances, well-known control structures and system components have not been shown in detail in order not to obscure the present invention.

In many instances, components implemented by the present invention are described at an architectural, functional level. Many of the elements may be configured using well-known structures, particularly those designated as relating to various compressed video signal processing techniques. Additionally, for logic to be included within the system of the present invention, functionality and flow diagrams are described in such a manner that those of ordinary skill in the art will be able to implement the particular methods without undue experimentation. It should also be understood that the techniques of the present invention may be implemented using a variety of technologies. For example, the various components of the splicing system described further herein may be implemented in software running on a computer system, or implemented in hardware utilizing microprocessors, application-specific integrated circuits,

programmable logic devices, or various combinations thereof. It will be understood by those skilled in the art that the present invention is not limited to any one particular implementation technique and those of ordinary skill of the art, once the functionality to be carried out by such components is described, will be able to implement the invention with various technologies without undue experimentation.

Referring now to Figure 1, there is illustrated a general purpose, multi-program splicing system 100. In this implementation, one or more programs are received on each of N different multiplexes 101a to 101N. A scheduling system 120 specifies which programs from each of the N input multiplexes will be selected for combination into M output multiplexes 110a to 110M. Changes to the formulation of each output multiplex 110 and the precise time of such changes are also controlled by the scheduling system 120.

Each output multiplex 110 may optionally be provided to one of several remultiplexers or reformatters 115a to 115M as shown in Figure 1. A remultiplexer 115 receives a program multiplex from the splicing system 100 and rearranges the packets constituting the several programs in order to more efficiently utilize the available channel bandwidth. An example of a suitable remultiplexer is described in copending U.S. Patent Application entitled "Method and Apparatus for Multiplexing Video Programs for Improved Channel Utilization," Serial No. 08/560,219, assigned to the assignee of the present invention and incorporated herein by reference. In alternative systems, the functions performed by a remultiplexer or reformatter 115 may be incorporated into the splicing system 100.

A more detailed illustration of the splicing system 100 is provided in Figure 2. Each incoming program multiplex

101 is received by one of N different input modules 210a - 210N. In addition to other functions which will be described more fully herein, the input modules 210 convert the signals received according to a particular interface to the format required for distribution over a common bus 215. The schedule translation unit 230 interprets the externally generated schedule or playlist from the scheduling system 120 to instructions for each of M output modules 220a - 220M. In this example, these instructions are also conveyed over the same common bus 215.

Each output module 220 has independent access to all programs received from each of the N input modules 210. The output modules 220 not only extract a particular set of programs from the bus 215 but also cut out old programs and cut in new programs into the output multiplex at various times as instructed by the Schedule Translation unit 230.

Embedded in each component data stream of each received MPEG program are time stamps specifying the precise time that video, audio, or other data frames are to be decoded and displayed at the receiver. In the case of video, some frames may be decoded at one time instant and displayed or presented at a later time instant. Such is the case for MPEG B-frames which depend on both preceding and following frame data to be properly decoded. For this reason, both Decoding Time Stamps (DTSs) and Presentation Time Stamps (PTSs) are sometimes included in the bit stream.

Time stamps alone are not sufficient to recover the timing of a program. Since the time stamps are related to a particular clock reference, and since the clock references used by different encoded programs may be independent, the MPEG standard requires that the current value of the clock reference be included in the bit stream at a certain minimum frequency. These clock reference values are referred to as PCRs.

One of the key tasks performed by an input module 210 is to normalize all time stamps (both DTS and PTS) so that they become relative to a single common reference. By insuring that the common reference has a known relationship with the current time of day (or some other reference shared with the scheduling system) it becomes possible to accurately identify the location within a data stream where a splice should be performed. In addition, it also becomes possible for a remultiplexer or reformatting system 115 to insure that the various frames of each program component are delivered to the receiving systems at a suitable rate. This results because the corrected time stamps now convey the information needed to insure that the corresponding data units arrive at the receiving station in time for decoding but not early enough to cause a buffer overflow .

An additional advantage of the time-stamp correction process is that it removes any further dependence on the PCRs embedded in the bit stream. This is necessary, since any buffering delays introduced during the course of processing the data streams of a program would seriously compromise the usefulness of the embedded PCRs. That is, the usefulness of a clock reference is decreased once the stream has been subjected to an unknown amount of delay. Processing elements having access to the same common reference need only consider the corrected time stamps when controlling playback timing. At some point, the PCRs must be regenerated based on the common reference. In this embodiment, the PCRs are regenerated by a remultiplexer unit 115.

A suitable remultiplexer 115 that can be adapted to use the corrected time stamps to control the rate of transmission of the various constituent streams is described in the above-referenced copending patent application. This adaptation is achieved by using the flowchart described further

herein with respect to Figure 14 for sequencing the stream of packets for transmission.

The process of normalizing time stamps to a single common reference is shown in Figure 3, which shows a more detailed version of an input module 210. A program multiplex is received through an input interface 310. A PCR counter 315 is maintained for each program in the received multiplex. (Different programs having a common PCR reference may share the same PCR counter 315). Since all time stamps encoded in the form specified by MPEG today are based on a 90 KHz clock, each counter in this example is incremented at this same rate. Each time a PCR is detected in the bit stream, it is immediately used to load the corresponding counter 315. Alternatively, a more sophisticated phase-locked loop counter solution may be employed.

In addition to the individual PCR counters, a shared system PCR counter 325 locked to the common system reference is also maintained. Each time a time stamp is received (DTS or PTS), it is summed with a correction value computed as the current difference between the counter value stored in the system PCR reference counter 325 and the count corresponding to the PCR reference for the particular program. The time stamp embedded in the bit stream is replaced with this result. The processes of PCR Extraction, Time Stamp Extraction, and Time Stamp Insertion are shown in more detail by the flow charts in Figures 4, 5, and 6 respectively. These flow charts are specific to the MPEG standard.

Figure 4 is a flow diagram for the process of PCR Extraction. In MPEG this is accomplished by extracting from the bitstream and decoding Program Association Tables (PATs) which identify the programs comprising the multiplex and provide the information needed to locate the Program Map Tables (PMTs),

which are also embedded in the same multiplex. Each PMT is associated with one of the multiplexed programs and provides the information needed to identify the video, audio, and any other components of the particular program. Specifically, this information is in the form of a Packet ID (PID). PIDs are embedded in the header of all packets and only one PID value is used to identify all packets of a single data stream component of the multiplex. The PMT also specifies the data stream containing the PCRs to be used for reconstructing the timing of the program. The PID corresponding to the PAT is always fixed, and therefore known, while the PIDs of the PMTs and the PIDs of the stream components of a program are usually inferred by reading the PAT and PMT tables respectively.

The procedure starts at Start Box 410 with a first step of getting the next packet from the incoming stream at step 420. If the packet is a PAT packet at step 430, then at step 440, the PMT PIDs for each program are extracted. If the packet was not a PAT packet, then at step 450 it is determined whether it is a known PMT packet. If it is, then at step 460, the PCR PID for the particular program is extracted, and the procedure returns to get the next packet. If the packet is not a known PMT packet at step 450, then at decision box 470 it is determined whether the packet is one with a PCR. If not, the procedure returns to get the next packet. If it is a packet with a PCR included, then at step 480 the PCR is extracted and at step 490 the PCR is copied to the output port assigned for the particular program. The procedure then returns to get the next packet again at step 420.

Figure 5 illustrates the procedure for Time Stamp Extraction which begins at start box 510 with a first step of getting a next incoming packet at step 520. If the packet is found to be a PAT packet at decision box 530, then at step 535, the PMT PIDs for each program are extracted. If the packet is

not a PAT packet at decision box 530, then at decision box 540 it is determined whether the packet is a known PMT packet. If it is, then at step 545 the PIDs for each stream component (video, audio and/or data) are extracted. If the packet is not a known PMT packet, then at decision box 550 it is determined whether or not the packet is a known component of one of the programs. If it is, then at decision box 560 it is determined if a Decoding Time Stamp (DTS) is present in the packet. If so, the DTS is copied to the output port assigned for the particular program at step 570. Then, it is determined at decision box 580 whether a PTS is present in the packet and, if so, at step 590 the PTS is copied to the output port assigned to the program.

Figure 6 illustrates the procedure for Time Stamp Insertion, which begins at start box 610 with a first step of retrieving the next packet at step 615. If the packet is a PAT packet at decision box 620, then the PMT PIDs for each program are extracted at step 625. If the packet is a known PMT packet at decision box 640, then the PIDs for each stream component of the particular program are extracted at step 645. If the packet is a known component of one of the programs at step 650, then it is determined at decision box 660 whether a DTS is present in the packet. If so, then at step 670, the DTS is replaced with a DTS received on an input port assigned to the particular program. Then, it is determined at decision box 680 whether a PTS is present in the packet. If so, the PTS is replaced with a PTS received from the input port assigned to the particular program at step 690. Before the next packet is retrieved at step 615 again, the packet is copied to output at step 630.

A more detailed block diagram of an output module 220 is shown in Figure 7. In the illustrated embodiment, each output module 220 combines a plurality of programs selected from the common bus 215, into a single multiplexed output signal 110 propagated through an output interface 715. A single Program

Splicer 720 processes the audio and video corresponding to each output program 725. The Program Splicer 720 receives the audio and video components of a *current program* and the audio and video components of a *next program* from a pair of demultiplexers 740 configured by the schedule control information received from the bus. The schedule control information originates from the scheduler (Figure 1) and includes the splice time for executing the next transition from the current program to the next program. After the transition has been completed, the next program becomes the current program and the scheduler sets a new next program and specifies the new splice time.

Seamless splicing is complicated by the particular sequence in which different types of video frames are received and reordered for display. Figure 8 illustrates an example from MPEG. The frames of both the current stream and the next stream are shown sequenced in the order in which they are received (decoding order). Also shown in Figure 8 is the resulting stream that has been spliced according to the methods of the current invention. This result is shown in both decoding and presentation order. Note that the presentation order is specified by the index assigned to each frame. This index is also known as the *temporal reference*. Since the B-frames cannot be reconstructed without knowledge of the anchor frames (the closest I- or P- frames both preceding and following a B-frame), both anchor frames must be sent first. In general, the decoding time and presentation time are the same for B-frames (PTS=DTS). However, an anchor frame is not presented before the next anchor frame is decoded. Until this time, all B-frames that are received must be decoded and presented first.

The first step in performing the splice is to compare the time stamps associated with each frame with the scheduled splice time. For maximum timing accuracy, the presentation time

stamp (PTS) should be used and the comparison test should only be performed when an I- or P- frame is received. This is because the stream should not be interrupted at a B-frame since a display artifact may occur if the receiver is sent an anchor frame, but not the in-between B-frame(s) that are to be displayed before it. It should be noted that MPEG does not require that all frames be accompanied by time stamps. This is because the missing time stamps can always be extrapolated from a previous time stamp based on the known frame rate and frame-display intervals.

Once the splicing point has been located, the next step is to insure that the next stream begins with an I-frame since only an I-frame can be decoded independently of other frames in the sequence. Generally, I-frames are accompanied by various headers containing information needed to decode the bit stream. All such headers are herein considered to be part of the I-frame. In certain cases, when transitioning from one stream to another stream having different decoding parameters, the first I-frame of the next stream may fail to include the headers necessary to convey the change in parameter settings. In that case, the splicing system should either insert a copy of the most recently received relevant header, or the splice can be deferred until such header information is received. It should also be noted that certain early-model MPEG receivers may be incapable of performing a seamless transition between two streams characterized by different encoding parameters such as those pertaining to image resolution. Other MPEG receivers may require the insertion of a 'Sequence End Code' into the compressed bitstream in order to execute the transition seamlessly. Therefore, it may be advantageous for the splicing system to insert the 'Sequence End Code' at the point of transition, either unconditionally or whenever certain changes in parameter settings are detected.

In addition to aligning the most recent I-frame in the next video stream with the splice transition instant, the splicer must delete all B-frames encountered after the I-frame and before the next I- or P-frame. Since B-frames are dependent on two anchor frames, of which only one is retained after the splice, these first B-frames cannot be successfully reconstructed.

Figure 9 is a block diagram of an exemplary Program Splicer 720 in accordance with the present invention. In this particular example, the Program Splicer includes two program processors 910a and 910b. At any given time, only one Program Processor is active and supplying video and audio data to OR gates 915 and 920 respectively. During this time, the other Program Processor remains in standby mode. The video and audio outputs of the processor that is in standby mode are set to logic '0' while the video and audio inputs are queued in preparation for the next splice. The time of the next splice is supplied by Schedule Control input signal 925. This signal is always monitored by the active Program Processor. It tells the active Program Processor when the current program should be cut out and the next program should be cut in. At the appropriate time, when the active Program Processor determines that the current program is at a suitable point for splicing, it generates a Splice Command output signal which is provided to the standby Program Processor. At the same time, the active Program Processor switches to standby mode and the standby Program Processor becomes active. Each time a Splice Command signal is received from either Program Processor it is provided by OR gate 930 to optional GOP Corrector 940 and to Time Stamp Corrector 950.

A more detailed block diagram of a particular implementation of the Program Processor 910 is shown in Figure 10. Video FIFO 1010 and audio FIFO 1015 are used to align the

output of a Program Processor with the output of the other Program Processor at the time of a splice. During active mode the FIFO control signals are static with the inputs and outputs enabled and the reset disabled. During standby mode, the FIFO control signals are determined by Latches 1020, 1025, 1030, and 1035. Latch 1035 specifies whether the processor is in standby or active mode. When in standby mode, Latch 1035 causes AND gates 1038 and 1039 to set audio and video outputs, respectively, to logic '0'. At the same time Latch 1035 will enable I-frame Detector 1040. Each time the beginning of an I-frame is detected on the video input signal, I-frame Detector 1040 will cause video FIFO 1010 to reset. At the same time, Latch 1025 becomes set, thereby allowing the I-frame data to be written into video FIFO 1010. I-Frame Detector 1040 also sets Latch 1020, thereby causing B-frame Detector 1045 to begin detecting B-frames. If the beginning of a B-frame is detected, then Latch 1025 will be cleared by B-Frame Detector 1045, thereby preventing the B-frame data from being written into video FIFO 1010. At the same time, Latch 1025 now causes I/P-Frame Detector 1050 to begin detecting both I-frames and P-frames. If the beginning of either type of frame is detected, then I/P-Frame Detector 1050 will again cause Latch 1025 to be set, allowing video FIFO 1010 to resume receiving data. I/P-Frame Detector 1050 also clears Latch 1020, thereby disabling the detection of B-frames by B-Frame Detector 1045. All video data is then written into video FIFO 1010, continuing until the next I-frame is detected by I-frame detector 1040. At this time, the video FIFO 1010 is again reset, and the initialization sequence repeats.

Each time video FIFO 1010 is reset, audio FIFO 1015 is reset by Latch 1030. The resetting of a FIFO essentially erases its contents. The FIFO will remain empty as long as it is in reset mode. The audio FIFO 1015 will remain in reset mode until

the beginning of a new audio frame is detected by Audio Frame Detector 1055. The Audio Frame Detector then clears Latch 1030, thereby allowing the detected audio frame to be written into audio FIFO 1015.

This process of resetting, and enabling/disabling the inputs of the audio and video FIFO's continues as long as the Program Processor remains in standby mode. Eventually, when a Splice Command input pulse causes Latch 1035 to become set, the Program Processor switches to active mode and the video output from FIFO 1010 and the audio output from FIFO 1015 begin to propagate through AND gates 1038 and 1039, respectively. The video data will begin with an I-frame and will contain no B-frames prior to the next following I- or P-frame. The video stream will therefore be decodable without any artifacts resulting from the absence of preceding data. Similarly, the audio stream will begin with a new audio frame closely aligned with the video.

Once the Program Processor enters active mode, the Next Splice input signal must be continuously compared to the time stamps embedded in the audio and video streams in order to permit the changeover back to standby. Audio time stamps are extracted by Time Stamp Extractor 1060. The extracted time stamp is then summed with the audio frame period to obtain the time of completion of the current audio frame. If Comparator 1065 determines that this frame completion time exceeds the scheduled time of the next splice, then it will cause AND gate 1038 to output a logic level of '0', effectively deleting the current audio frame. This step is taken to insure that the splice occurs at audio frame boundaries without overlapping the audio segment of the current and next programs. In this preferred case, a short silent audio interval is accepted in return for a seamless video transition.

The video time stamps are extracted by Time Stamp Extractor 1070 and compared to the next splice time by Comparator 1075. If an extracted time stamp is greater than or equal to the next splice time, then Comparator 1075 enables I/P frame Detector 1080. Upon detection of the beginning of the next I- or P-frame, I/P Detector 1080 issues a pulse on the Splice Command output signal. This pulse clears Latch 1035 and causes the Program Processor to return to standby mode. The Splice Command output pulse is also coupled to the Splice Command input signal of the other Program Processor, thereby allowing the other Program Processor to complete the splice.

One method for implementing Time Stamp Extractors 1060 and 1070 was described with reference to Figure 5. Methods for implementing Frame Detectors 1040, 1045, 1050, 1055, and 1080 are well known to those familiar with the MPEG standard. The process of generating the time-shifted and edited sequence of audio and video frames for output by the Program Processor 910 is illustrated by the timing diagrams shown in Figure 11. The signals correspond to the operation of the circuit described with respect to Figure 10.

The Program Processor shown in Figure 9 is an example of a hardware implementation. Redundant active and standby Program Processors have been described in order to simplify the presentation of a complete solution capable of multiple and repetitive splices. However, it should be noted that the two-processor solution could be replaced by a single integrated processor with similar logic for processing the current program in active mode and the next program in standby mode. Then, upon receiving a splice command, the next stream would be switched to the current program logic and a new next program would be switched to the next program logic. This single-processor alternative is particularly well suited for software solutions

which are typically implemented as sequential processes rather than multiple concurrent ones.

Once the current- and next-program streams have been merged, certain embedded bit-stream parameters must be adjusted in order to reflect some of the changes that have occurred. One of these parameters is the temporal reference. This is because MPEG specifies that the temporal reference of the first frame to be displayed in each Group of Pictures (GOP) be assigned a temporal reference of zero. Assume, for example, that the next program following the splice begins with a new GOP. Then, using the example of Figure 8, the temporal references must be decremented by two since the two deleted B-frames received after the I- frame were intended to be the first two *displayed* frames of the GOP. This adjustment of the temporal references ceases once the next GOP header is received. The process is illustrated by the flow chart in Figure 12.

The process begins at start box 1210 once a Splice Command signal is detected. Then the first step 1220 is to extract the temporal reference for the next frame (I-frame). At step 1230, a variable, TRc, is set equal to the extracted temporal reference of the I-frame. Then at step 1240, the temporal reference in the bit stream is changed to zero. If, at decision box 1250, it is determined that the GOP header is detected, then a closed-GOP bit is set at step 1260. Then, or otherwise, at step 1270, the procedure waits for a next frame. If the next frame does not include a GOP header at decision box 1280, then at step 1290, the value of TRc is subtracted from the temporal reference in the bitstream and the procedure returns to step 1270 to wait for a next frame. When a GOP header is detected at decision box 1280, the procedure ends.

A closed GOP is a group of pictures wherein each of the frames comprising the group can be regenerated without

referencing any frames outside of the group. If the program-splicing process is implemented according to the method and hardware depicted in Figure 9 and according to the example of Figure 8, then the first GOP after the splice point will always become closed even if it was initially open. As shown in the flowchart of Figure 12, the *closed-GOP* bit is set accordingly.

Although the adjustments of the temporal reference and closed-GOP parameters are necessary to retain compliance with the MPEG standard, these parameters are likely to be ignored by a large percentage of existing MPEG decoders. Instead, these decoders will rely on time stamps to control the sequencing of the displayed frames. In the absence of time stamps, each decoder infers the temporal reference based on standardized rules for the decoding and display of MPEG sequences. The parameters which cannot be ignored, however, are the decoding and presentation time stamps, and these parameters must also be adjusted whenever different programs are spliced. That is, the same time stamps that were adjusted by the input modules 210 to compensate for variations among different clock references must now be adjusted again, this time to compensate for the time shift incurred by aligning the I-frame of the next stream with the splice point of the current stream.

A method for performing the second Time Stamp Correction 1300 is described with respect to the flow chart in Figure 13. A correction value corresponding to the time duration of the delay incurred at the video FIFO 1010 (Figure 10) is computed at steps 1335, 1340 and 1345 upon receipt at step 1330 of a first frame after the splice. DTS_1, determined at step 1335, is the decoding time stamp which would have been assigned to the corresponding frame of the previous stream if the splice had not occurred. DTS_2, computed at step 1340, is the decoding time stamp that was previously assigned to the first frame of the next stream prior to the delay introduced by video

FIFO 1010. The difference between these two time stamps is the correction value TS_c. This correction value is then added to all of the PTS (step 1385)and DTS (step 1375) time stamps which follow. The exception at step 1360 is the PTS of the I-frame used to initiate the next stream immediately after the splice. By comparing the decoding order and presentation order versions of the resulting spliced stream for the example of Figure 8, it may be observed that, except for the one exception, the I- and P-anchor frames are presented exactly three frame intervals after they are decoded. The delay is to account for the in-between B-frames and the subsequent frame reordering. However, since the first two frames received after the splice are both anchor frames, and there are no B-frames to be shown between them, the reordering delay is reduced. In this case the I-frame (the first of the two anchor frames) would be displayed immediately after displaying the last frame ($X8_P$) of the previous stream. Since the presentation time of this last frame is equal to the decoding time of the I-frame, the I-frame PTS can be derived by adding the display interval time (calculated at step 1325) of the preceding last frame to the I-frame DTS at step 1360. In addition, since display interval times may vary depending on whether the sequence is encoded in field or frame mode, and whether a film-to-video conversion is being implemented, the display interval should be determined by examining the codewords in the bit stream which contain this relevant information (specifically the MPEG frame rate value, the MPEG field/frame encoding mode, and the MPEG repeat-first-field flag).

Another problem may occur if splicing is permitted to occur not only at frame boundaries but at field boundaries as well. Although frames are conventionally defined as a pair of two fields where one field comprises the odd lines and the other comprises the even lines, the MPEG-2 definition is less strict.

In MPEG-2, the field which occurs earlier in time may be either the field with the odd lines or the field with the even lines. A specific bitstream flag (known as top-field-first) is provided to make this distinction. In addition, an MPEG-2 frame is permitted to span not only two field intervals, but occasionally three field intervals. However, instead of encoding the third field, a flag is set (known as repeat-first-field) to indicate to the decoder that a third field is to be displayed and that it should be an exact copy of the first field. Note that the second field could not have been repeated instead, since consecutive fields must alternate between odd lines and even lines to avoid artifacts.

Assume, for example, that the current program ends with a field consisting of even lines and the next program begins with a field also consisting of even lines. A spliced bitstream that results in two successive fields both with even (or odd) lines is not permitted by MPEG, as such streams cannot be displayed correctly by existing display devices. Different steps can be taken to prevent this situation. The preferred solution is to edit the repeat-first-field flag corresponding to the last frame of the current stream prior to the splicing point. If the last displayed field of the current stream and the first displayed field of the next stream have the same odd/even line parity, then the repeat-first-field flag can be simply toggled. That is, a value of logic '0' could be changed to a logic '1' while a logic '1' could be changed to a logic '0'. The effect is to lengthen or shorten respectively the last frame of the current stream by exactly one field. In some cases, special steps may need to be taken to insure that this variation is properly accounted for when determining the time stamp correction factor (TS_c).

In some implementations, it may be more advantageous to implement some of the program splicing subprocesses in the

remultiplexer/reformatter 115 instead of in the output modules 220 of the splicing system 100. For example, one of the tasks of the remultiplexer may be to limit or regulate the data rate of the resultant multiplex. This data rate regulation capability greatly improves the versatility of the switching system and allows the use of variable bit rate (VBR) streams. A good statistical remultiplexer can efficiently combine multiple VBR streams into a single multiplex, but depending on the particular schedule in effect at any given time, the combined data rate may exceed the available capacity of the channel, in which case some data rate regulation capability will be needed. In such cases, it may be easier to implement some of the post-splice adjustments after the date rate has been limited to a more manageable rate. As an example, a software-based remultiplexer may have sufficient reserve processing capacity to implement most if not all of the stream splicing functions once the data rate has been limited.

An example of a remultiplexer 115 with data rate regulation capability is described in copending U.S. Patent Application Serial No. 08/631,084 entitled "Compressed-Video Reencoder System for Modifying the Compression Ratio of Digitally Encoded Video Programs," assigned to the assignee of the present invention. Such a remultiplexer may be advantageously coupled to the output of an output module for a second reason. This is because the remultiplexer can insure that the receiver buffer occupancy level remains within limits after switching between two different streams. As discussed previously, the buffer could otherwise overflow or underflow even when both streams have been encoded at the same data rate. This is due to the different buffer occupancy levels assumed by the two encoders, corresponding to the current and next streams, at the time of the splice. The process for determining which

packet to send next, based on the fullness of the buffers at the receivers, is illustrated in Figure 14.

The procedure 1400 for determining which packet to send next first determines, at decision box 1410, whether all receiver buffers are already full. If they are, then a null data packet is sent at step 1420. Otherwise, a data packet is selected for the stream which has the smallest DTS for its next frame at step 1430. If sending this data packet would overflow the receiving buffer (decision box 1440), then at step 1450, the packet corresponding to the stream with the next smallest DTS for the next frame is selected. This process continues until a packet is selected that does not overflow the receiver's buffer. At that point, the next packet for the selected stream is transmitted at step 1460.

A second implementation variation would be to merge the remultiplexer 115 and output module 220 into a single unit.

There has thus been described a method and system for the seamless splicing of compressed video programming. Although the present invention has been described with respect to certain exemplary and implemented embodiments, it should be understood that those of ordinary skill in the art will readily appreciate various alternatives to the present invention. Accordingly, the spirit and scope of the present invention should be measured by the terms of the claims which follow.

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A splicing system for splicing a second compressed program to follow a first compressed program comprising:

means for receiving information identifying a scheduled splice time;

means responsive to said scheduled splice time for outputting said first compressed program and determining an appropriate location in said first compressed program for executing a splice transition, said means then providing a splice transition signal; and

means responsive to said splice transition signal for outputting said second compressed program at a permissible splice point therein so that said second compressed program splices to follow said first compressed program in a seamless manner to form a spliced data stream;

wherein said means responsive to said splice transition signal includes means for deleting bidirectionally-coded frames immediately following an intra-coded-frame at the permissible splice point and before a next anchor frame where the bidirectionally-coded frames depend on previous data that is not retained in the spliced data stream.

2. The splicing system of claim 1 further comprising time stamp normalization circuitry for normalizing time stamps associated with said first and second compressed programs.

3. The splicing system of claim 2 wherein said time stamp normalization circuitry comprises:

PCR extraction circuitry for extracting program clock references from said first and second compressed

programs as they are received;

first and second PCR counters synchronized with the PCR's received from the PCR extraction circuitry;

a system PCR counter synchronized with a local or externally received system clock reference;

time stamp extraction circuitry for extracting time stamp information from said first and second compressed programs;

time stamp adjustment circuitry for calculating a correction value for the time stamps for said first and second compressed programs equal to the difference between the count value of the system PCR counter and the count value of the associated PCR counter; and

time stamp insertion logic for inserting the adjusted time stamps back into the first and second compressed programs.

4. The splicing system of claim 2 wherein said means responsive to said splice transition signal includes:

means for delaying a video input signal corresponding to said second compressed program;

means for delaying one or more audio input signals corresponding to said second compressed program; and

means for outputting said delayed video and audio signals upon receiving a splice transition signal.

5. The splicing system of claim 4 wherein said delay means comprises a FIFO and FIFO control logic.

6. The splicing system of claim 5 further comprising frame detection circuitry for detecting intra-coded frames (I-frames) in said video input signal, said video frame detection circuitry coupled to cause said video FIFO logic to reset upon detection of I-frame data.

7. The splicing system of claim 6 further comprising:

first video frame detection circuitry for detecting bidirectionally coded frames (B-frames);

second video frame detection circuitry for detecting anchor frames (I- or P-frames); and

means for deleting any B-frames detected after an I-frame and before the next anchor frame.

8. The splicing system of claim 6 wherein said video frame detection circuitry is coupled to cause said audio FIFO to reset upon detection of I-frame data, and said splicing system further comprises audio frame start detection circuitry coupled to receive said audio input signal and coupled to said audio FIFO for keeping said audio FIFO in reset mode until detection of the beginning of a next audio frame.

9. The splicing system of claim 6 further comprising circuitry for adjusting embedded bit stream parameters including the time stamps in said second compressed digital program.

10. A method of splicing a second compressed program to follow a first compressed program in a seamless manner, said method comprising the steps of:

receiving a scheduled splice time;

determining a point in said first compressed program appropriate for executing a splice transition close to said scheduled splice time;

providing a splice transition signal: and

outputting said second compressed program beginning with an intra-coded video frame (I-frame) of said second compressed program in response to the splice transition signal to form a seamlessly spliced data stream;

wherein said outputting step includes a step of deleting bidirectionally-coded frames immediately following the I-frame and before a next anchor frame where the bidirectionally-coded frames depend on previous data that is not retained in the spliced data stream.

11. The method according to claim 10 further comprising the step of normalizing time stamps associated with said first and second compressed programs to a common reference.

12. The method according to claim 11 wherein said normalizing step comprises the steps of:

extracting program clock reference (PCR) signals associated with said compressed digital programs;

maintaining a system PCR signal;

extracting time stamps from said compressed programs as they are received;

offsetting said time stamps by a correction value equal to the current difference between the associated program PCR count and the system PCR count; and

inserting the adjusted time stamps back into said compressed digital programs.

13. The method according to claim 11 wherein said step of determining an appropriate point comprises the steps of:

extracting each time stamp from said first compressed program;

comparing said time stamps to said scheduled splice time;

detecting anchor frames in said first compressed digital program; and

providing a splice transition signal upon detection of the closest anchor frame to said scheduled splice time.

14. The method according tc claim 13 wherein said providing step instead occurs upon detection of the first anchor frame after said scheduled splice time.

15. The method according to claim 13 wherein said outputting step comprises the steps of:

buffering in a video data FIFO said second compressed program;

monitoring said second compressed program for intra-coded frames (I-frames) of encoded video data;

upon detecting an I-frame, resetting said video data FIFO: and

outputting said second compressed program upon detection of said splice transition signal.

16. The method according to claim 15 further comprising the steps of:

buffering in an audio data FIFO audio data associated with said second compressed program data;

resetting said audio data FIFO beginning when said video data FIFO is reset;

detecting the start of a next audio frame of data following the detection of each intraframe of data: and

ending the resetting of said audio FIFO upon such detection.

17. The method of claim 15 further comprising the steps of detecting and deleting any bidirectionally-coded frames (B-frames) after said I-frame and before the next anchor frame.

18. The method of claim 15 further comprising the steps of readjusting any embedded bitstream parameters that may have been changed by the processing of said second compressed program.

19. The method of claim 18 wherein said adjusting step includes the step of adjusting the temporal reference of the first group of frames following the splice point.

20. The method of claim 18 wherein said adjusting step includes setting the closed-GOP bit for the first group of frames following the splice point in said second compressed program.

21. The method of claim 18 wherein said adjusting step includes repeating a field when the current program and the next program have the same field parity at the time of the splice.

22. The method of claim 18 wherein said adjusting step including deleting a field when the current program and the next program have the same field parity at the time of the splice.

23. The method of claim 18 wherein said adjusting step includes the step of adjusting the time stamps associated with said second compressed program to compensate for the time shift resulting from aligning the second compressed program with the splice point of the first compressed program.

24. The method according to claim 23 wherein said time stamp adjustment step comprises the steps of:

calculating a correction value corresponding to the

time duration of delay incurred at the video data FIFO;

and adding said correction value to all of said time stamps in said second compressed program following said splice transition signal.

25. The method according to claim 24 further comprising the step of adjusting the presentation time stamp (PTS) corresponding to the first intra-coded frame (I-frame) following the splice point by adding a frame display interval time to the decoding time stamp (DTS) of the last anchor frame preceding the splice point.

26. A method of seamlessly splicing first and second compressed video programs, comprising the steps of:

shifting a scheduled splice time to coincide with a closest appropriate splice point of the first compressed program;

delaying the second compressed program until a permissible splice point therein becomes aligned with the shifted scheduled splice time;

cutting out the first compressed program at the appropriate splice point; and

cutting in the second compressed program at the permissible splice point to follow the first compressed program tc form a seamlessly spliced data stream;

wherein the step of cutting in the second compressed program includes a step of deleting bidirectionally-coded frames immediately following an intra-coded-frame at the permissible splice point and before a next anchor frame where the bidirectionally-coded frames depend on previous data that is not retained in the spliced data stream.

27. The method of claim 10 wherein the spliced data

stream is provided to a reformatter for adjusting the data rate of the spliced data stream to prevent overflowing or underflowing the rate buffer at a receiver.

28. The method of claim 10 wherein the spliced data stream is multiplexed with other program signals at a rate which prevents overflowing or underflowing the rate buffer at a receiver.

FIG. 1

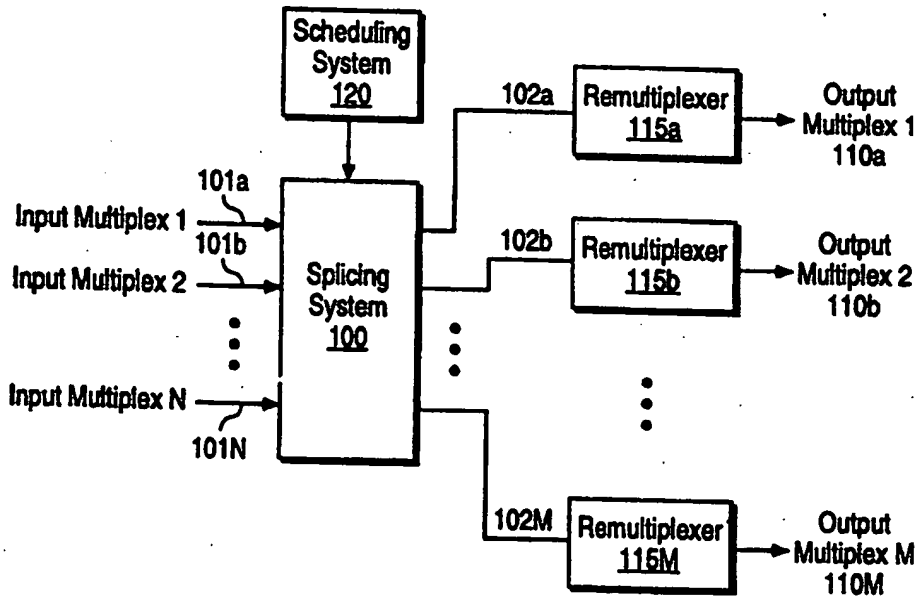
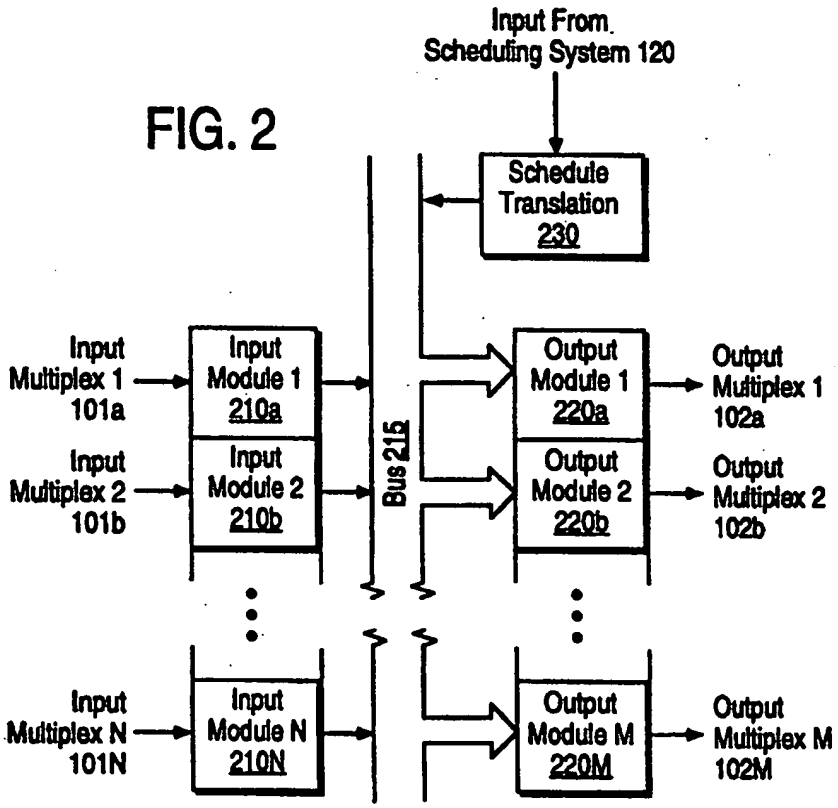

Scheduling System 120
Splicing System 100
Input Multiplex 1
Input Multiplex 2
Input Multiplex N
101a
101b
101N
102a
102b
102M
Remultiplexer 115a
Remultiplexer 115b
Remultiplexer 115M
Output Multiplex 1 110a
Output Multiplex 2 110b
Output Multiplex M 110M

FIG. 2


Input From
Scheduling System 120
Schedule
Translation
230
Input
Multiplex 1
101a
Input
Module 1
210a
Input
Multiplex 2
101b
Input
Module 2
210b
Bus 215
Output
Module 1
220a
Output
Multiplex 1
102a
Output
Module 2
220b
Output
Multiplex 2
102b
Input
Multiplex N
101N
Input
Module N
210N
Output
Module M
220M
Output
Multiplex M
102M

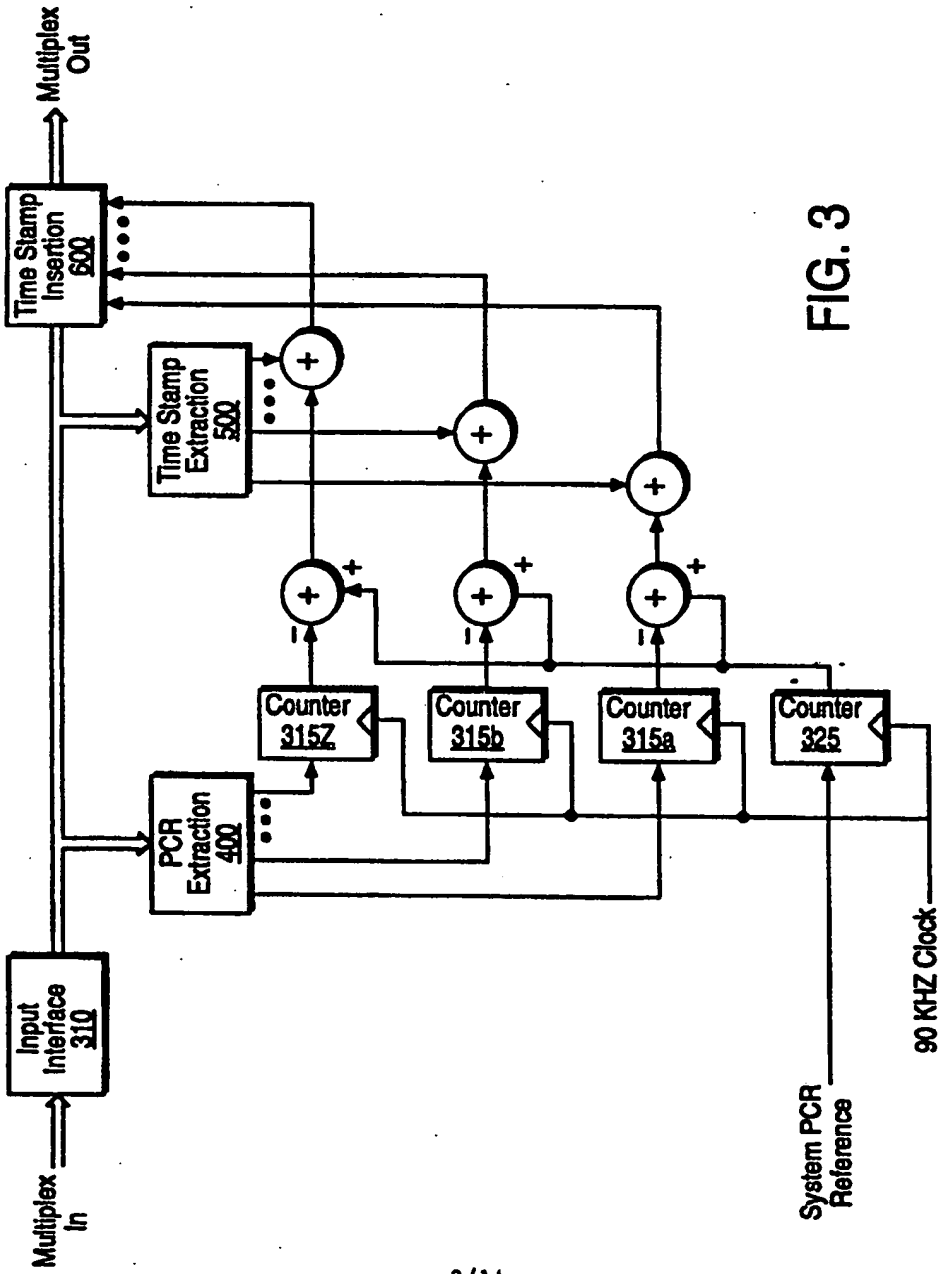

Multiplex Out
Time Stamp Insertion 600
Time Stamp Extraction 500
PCR Extraction 400
Input Interface 310
Multiplex In
Counter 315Z
Counter 315b
Counter 315a
Counter 325
System PCR Reference
90 KHZ Clock

FIG. 3

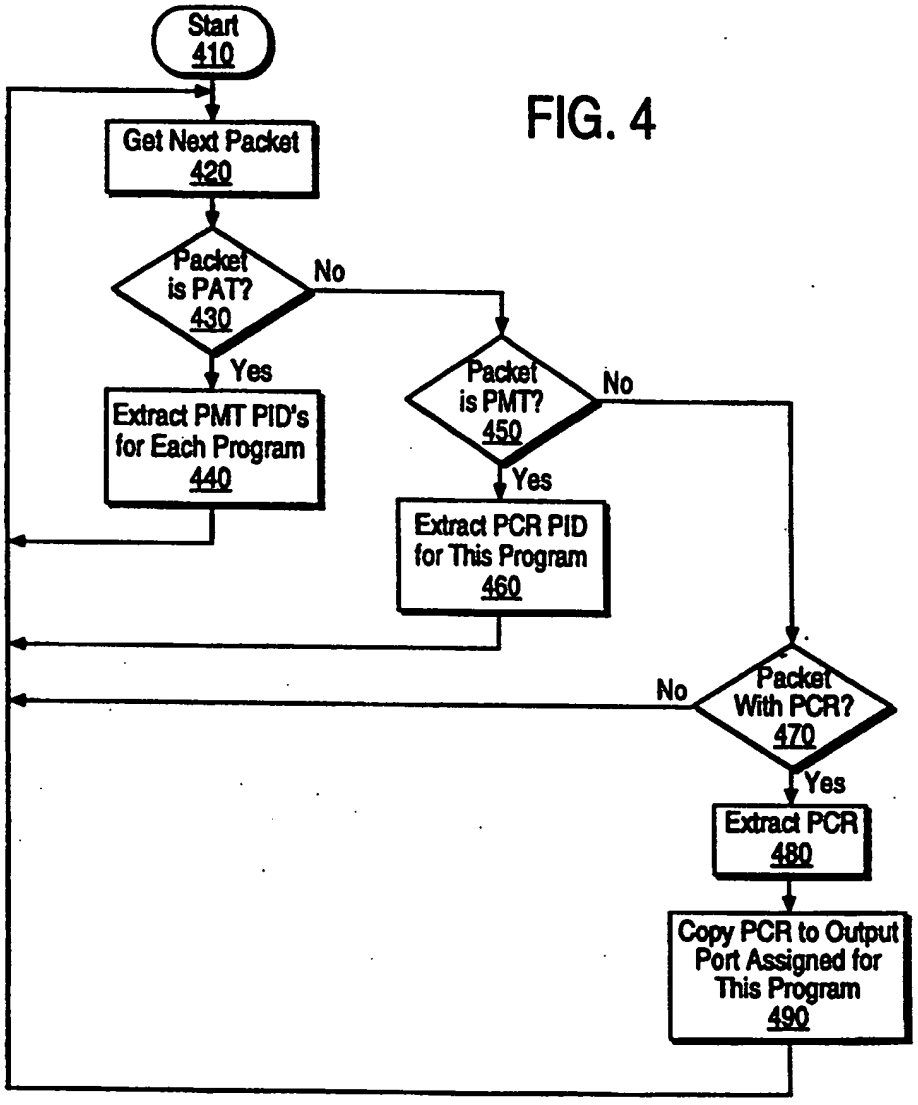

Start
410
Get Next Packet
420
Packet is PAT?
430
No
Yes
Extract PMT PID's for Each Program
440
Packet is PMT?
450
No
Yes
Extract PCR PID for This Program
460
Packet With PCR?
470
No
Yes
Extract PCR
480
Copy PCR to Output Port Assigned for This Program
490

FIG. 4

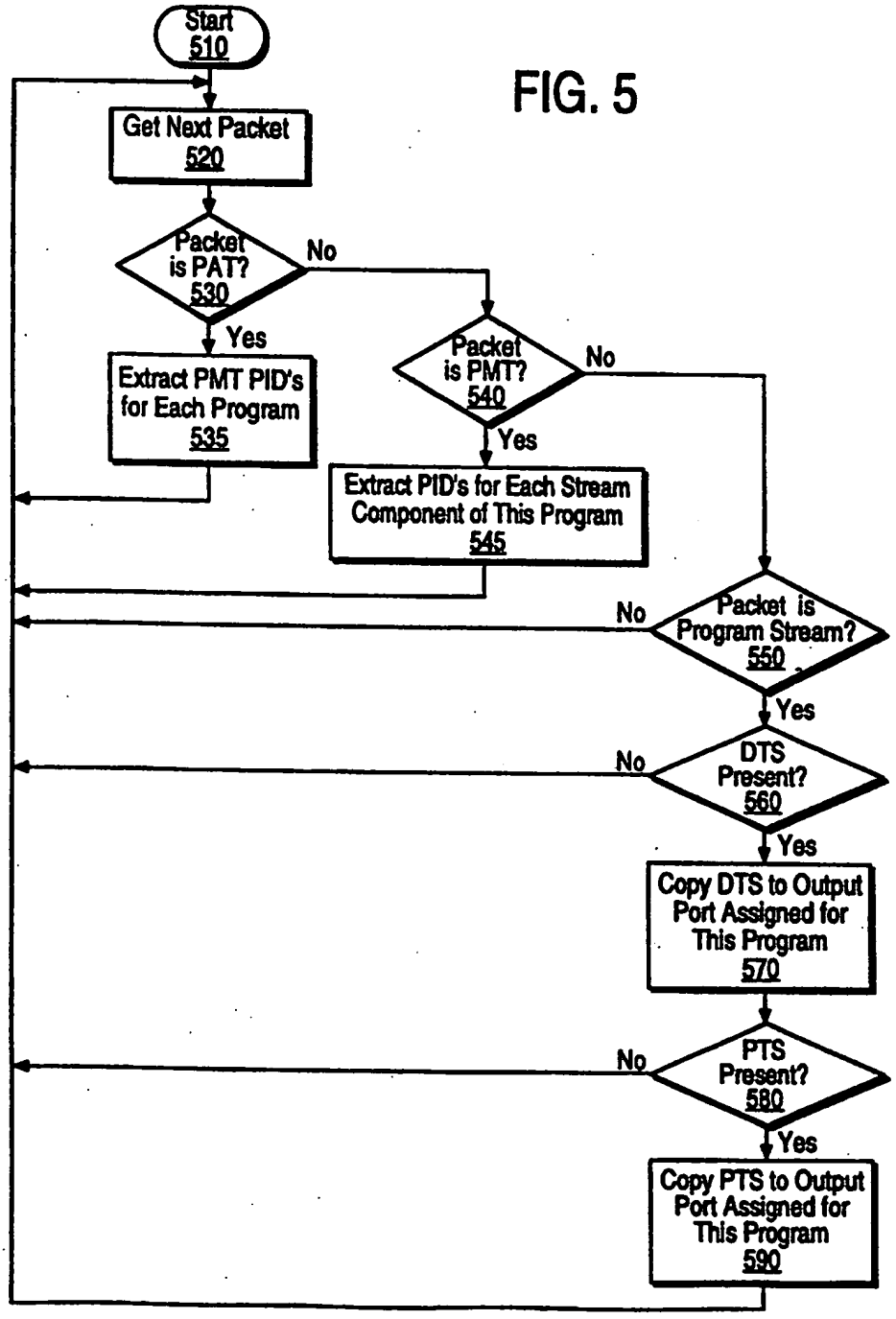
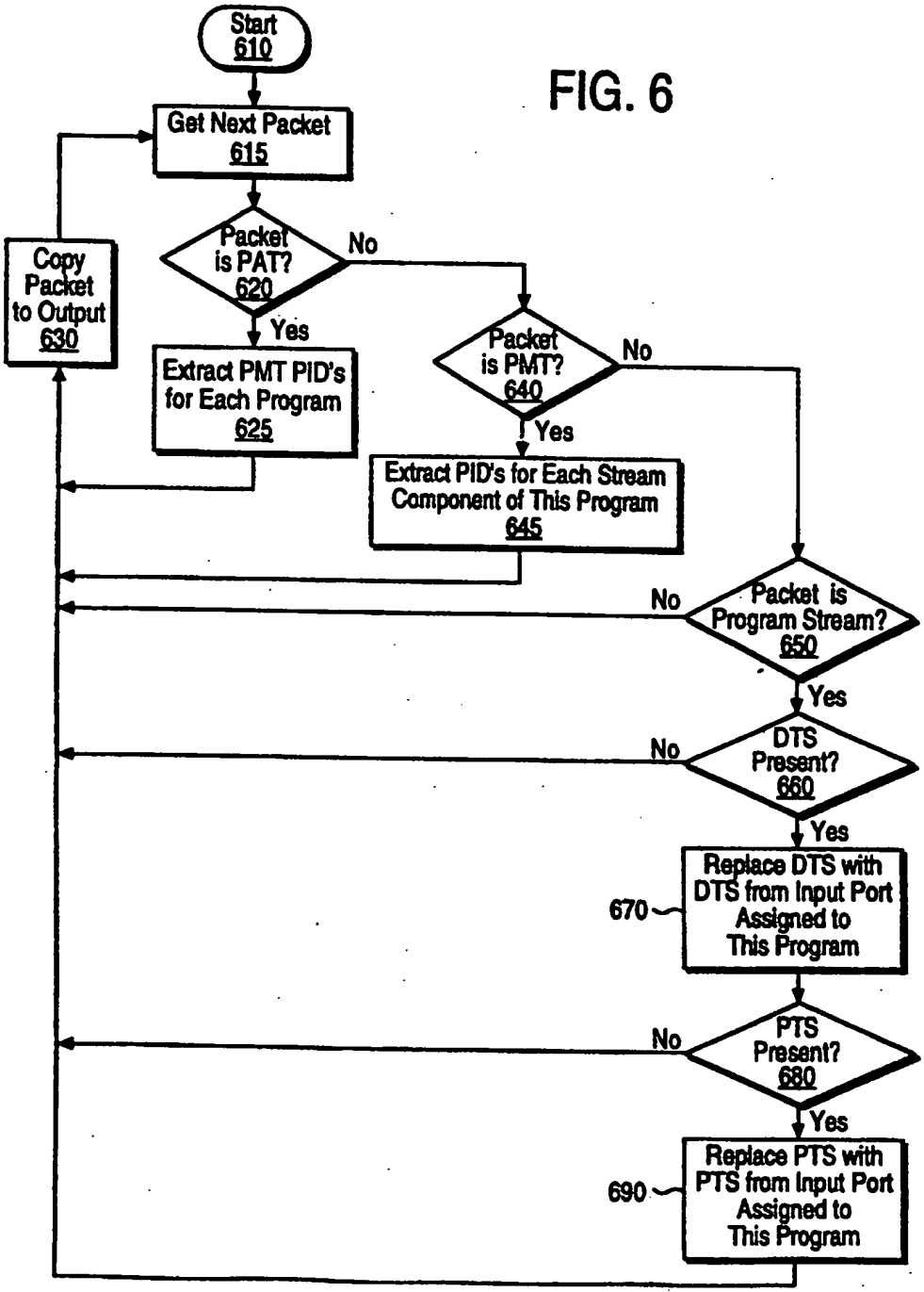

Start
510
Get Next Packet
520
Packet is PAT?
530
No
Yes
Extract PMT PID's for Each Program
535
Packet is PMT?
540
No
Yes
Extract PID's for Each Stream Component of This Program
545
Packet is Program Stream?
550
No
Yes
DTS Present?
560
No
Yes
Copy DTS to Output Port Assigned for This Program
570
PTS Present?
580
No
Yes
Copy PTS to Output Port Assigned for This Program
590

FIG. 5

FIG. 6


Start
610
Get Next Packet
615
Packet is PAT?
620
No
Yes
Copy Packet to Output
630
Extract PMT PID's for Each Program
625
Packet is PMT?
640
No
Yes
Extract PID's for Each Stream Component of This Program
645
Packet is Program Stream?
650
No
Yes
DTS Present?
660
No
Yes
Replace DTS with DTS from Input Port Assigned to This Program
670
PTS Present?
680
No
Yes
Replace PTS with PTS from Input Port Assigned to This Program
690

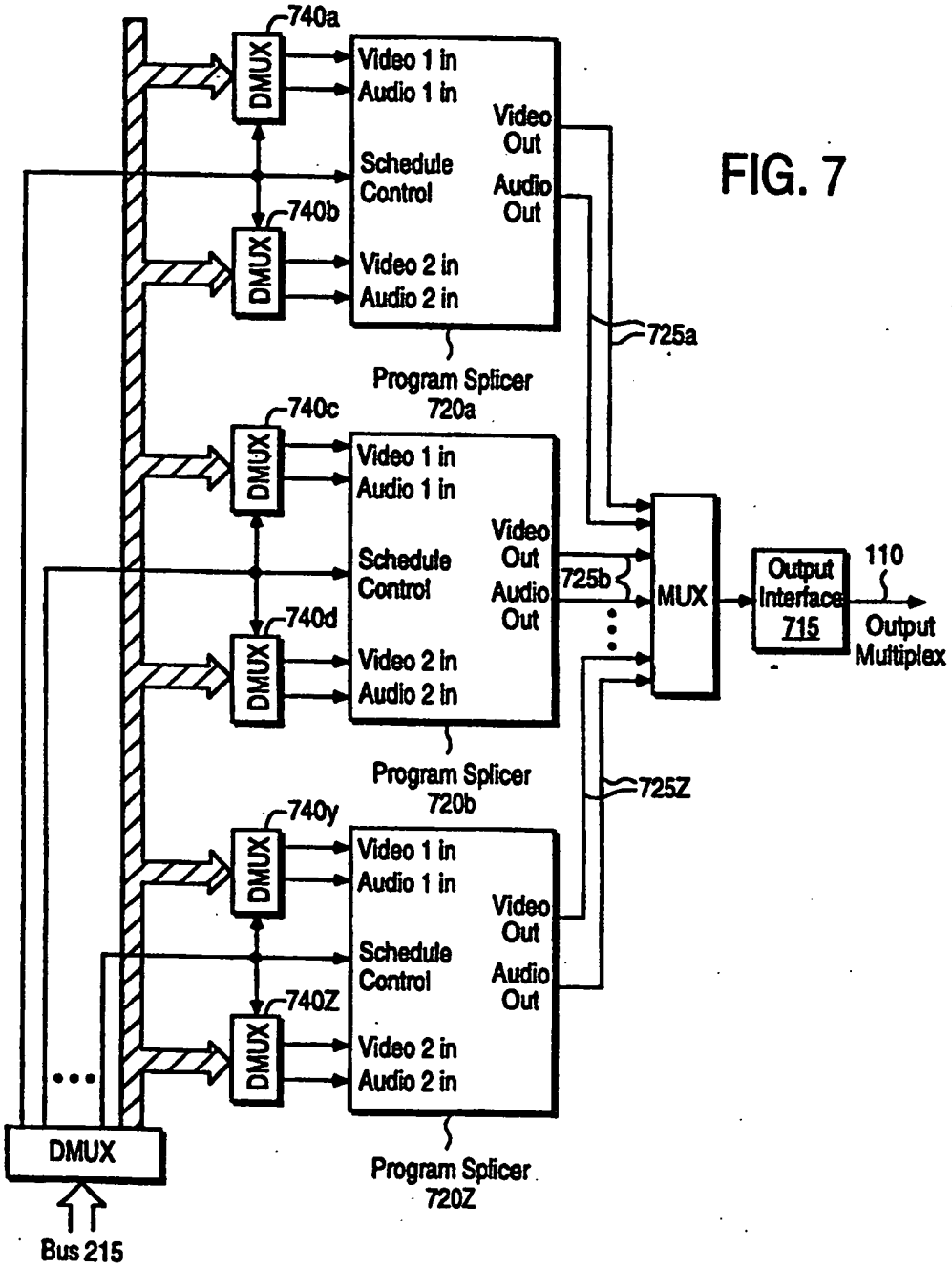
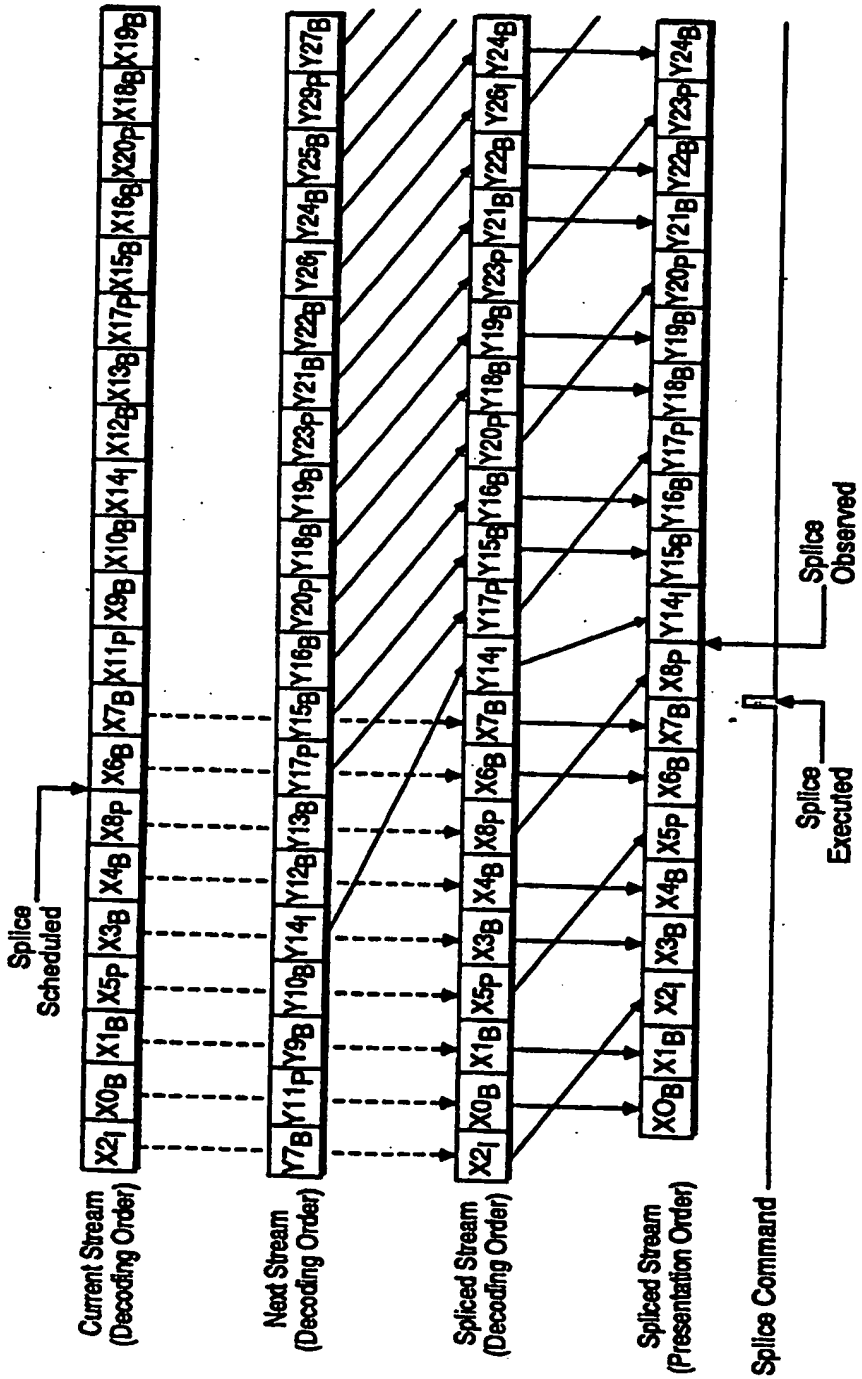

740a
DMUX
Video 1 in
Audio 1 in
Video Out
Schedule Control
Audio Out
740b
DMUX
Video 2 in
Audio 2 in
Program Splicer 720a
725a
740c
DMUX
Video 1 in
Audio 1 in
Video Out
Schedule Control
Audio Out
725b
740d
DMUX
Video 2 in
Audio 2 in
Program Splicer 720b
725Z
740y
DMUX
Video 1 in
Audio 1 in
Video Out
Schedule Control
Audio Out
740Z
DMUX
Video 2 in
Audio 2 in
Program Splicer 720Z
DMUX
Bus 215
MUX
Output Interface 715
110
Output Multiplex

FIG. 7

FIG. 8


Splice Scheduled
Current Stream (Decoding Order)
Next Stream (Decoding Order)
Spliced Stream (Decoding Order)
Spliced Stream (Presentation Order)
Splice Command
Splice Executed
Splice Observed

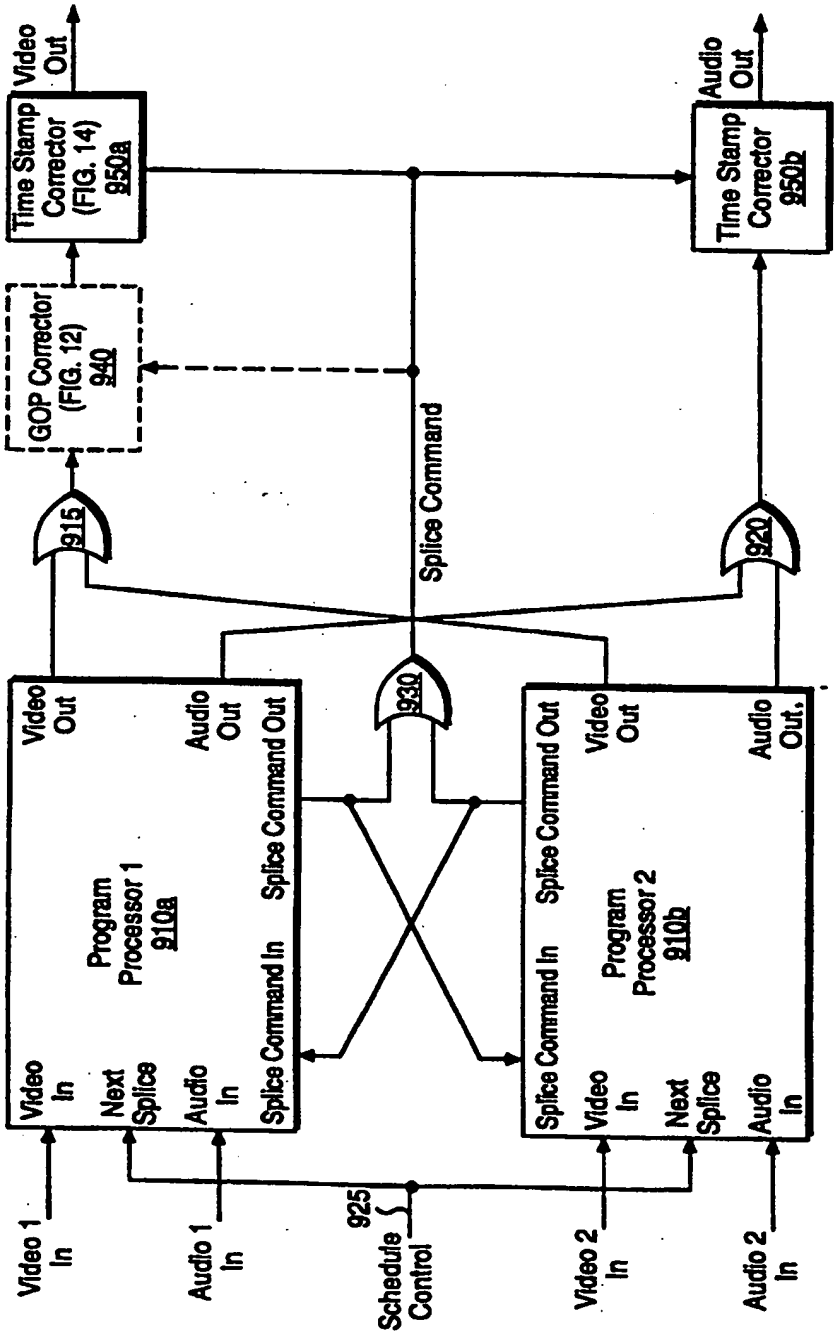

Video Out
Time Stamp Corrector (FIG. 14) 950a
Audio Out
Time Stamp Corrector 950b
GOP Corrector (FIG. 12) 940
Splice Command
915
920
930
Video Out
Audio Out
Splice Command Out
Program Processor 1 910a
Splice Command In
Video In
Next Splice
Audio In
Splice Command Out
Video Out
Audio Out,
Program Processor 2 910b
Splice Command In
Video In
Next Splice
Audio In
Video 1 In
Audio 1 In
925 Schedule Control
Video 2 In
Audio 2 In

FIG. 9

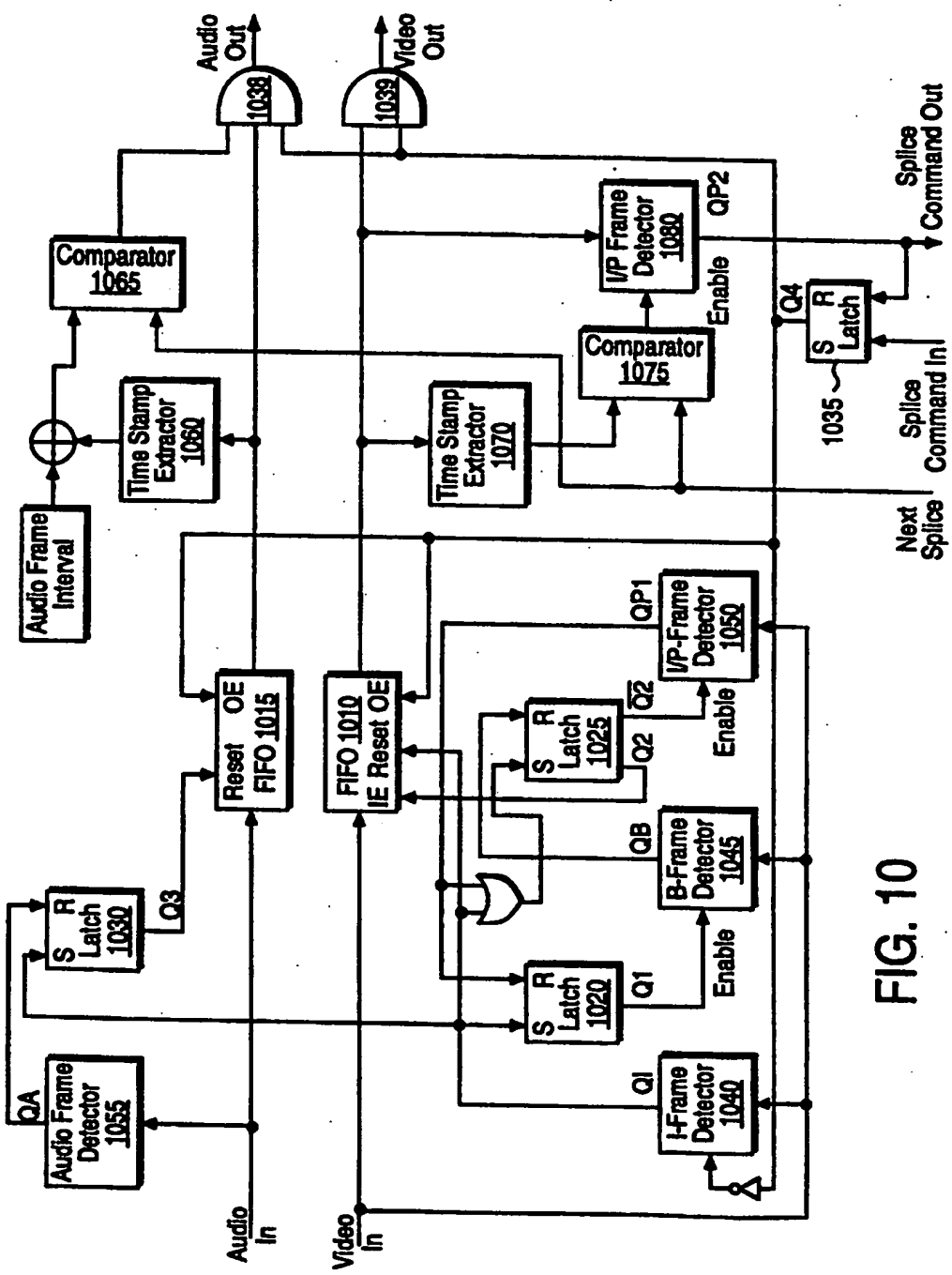

Audio Out
1038
Video Out
1039
Comparator 1065
Time Stamp Extractor 1060
Audio Frame Interval
Time Stamp Extractor 1070
I/P Frame Detector 1080
QP2
Enable
Comparator 1075
Q4
S R Latch
1035
Splice Command Out
Splice Command In
Next Splice
QP1
I/P-Frame Detector 1050
Reset OE FIFO 1015
FIFO 1010 IE Reset OE
S R Latch 1025
Q2
Enable
QB
B-Frame Detector 1045
S R Latch 1030
Q3
S R Latch 1020
Q1
Enable
QA
Audio Frame Detector 1055
QI
I-Frame Detector 1040
Audio In
Video In

FIG. 10

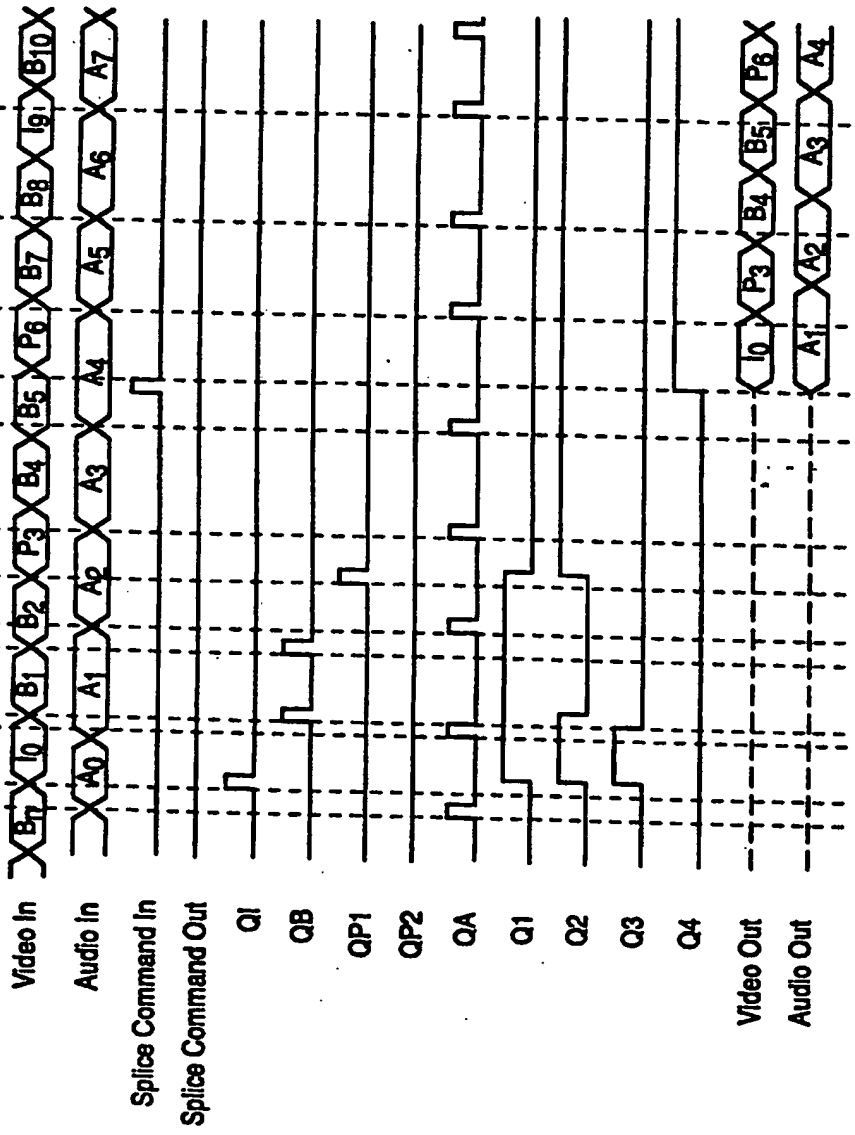

Video In
Audio In
Splice Command In
Splice Command Out
QI
QB
QP1
QP2
QA
Q1
Q2
Q3
Q4
Video Out
Audio Out

FIG. 11

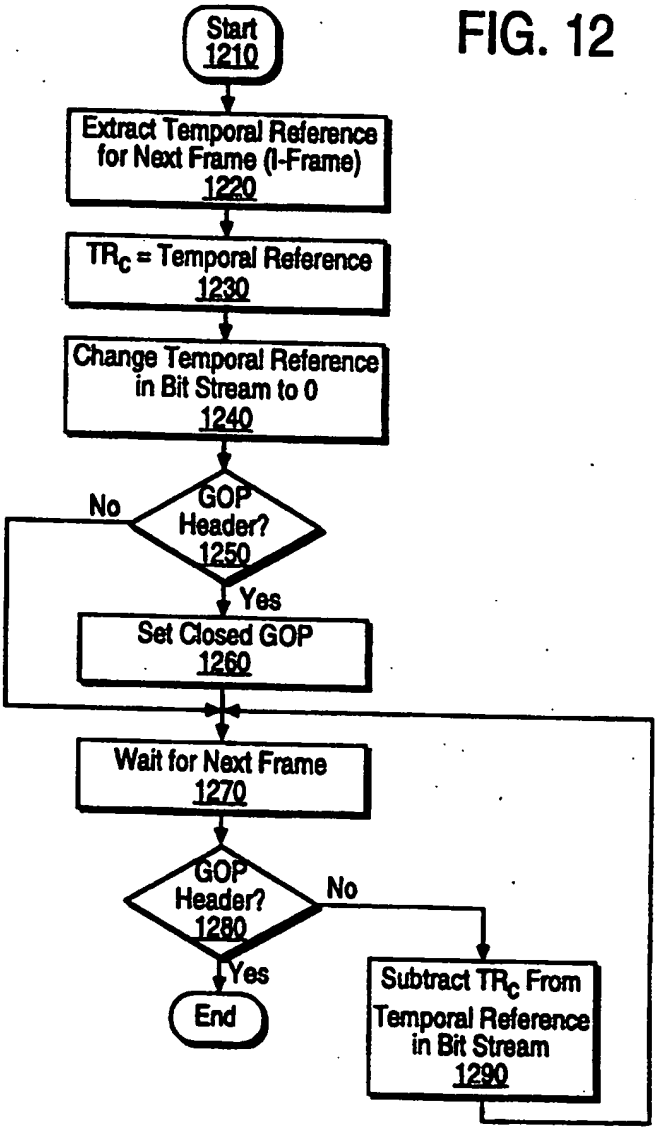
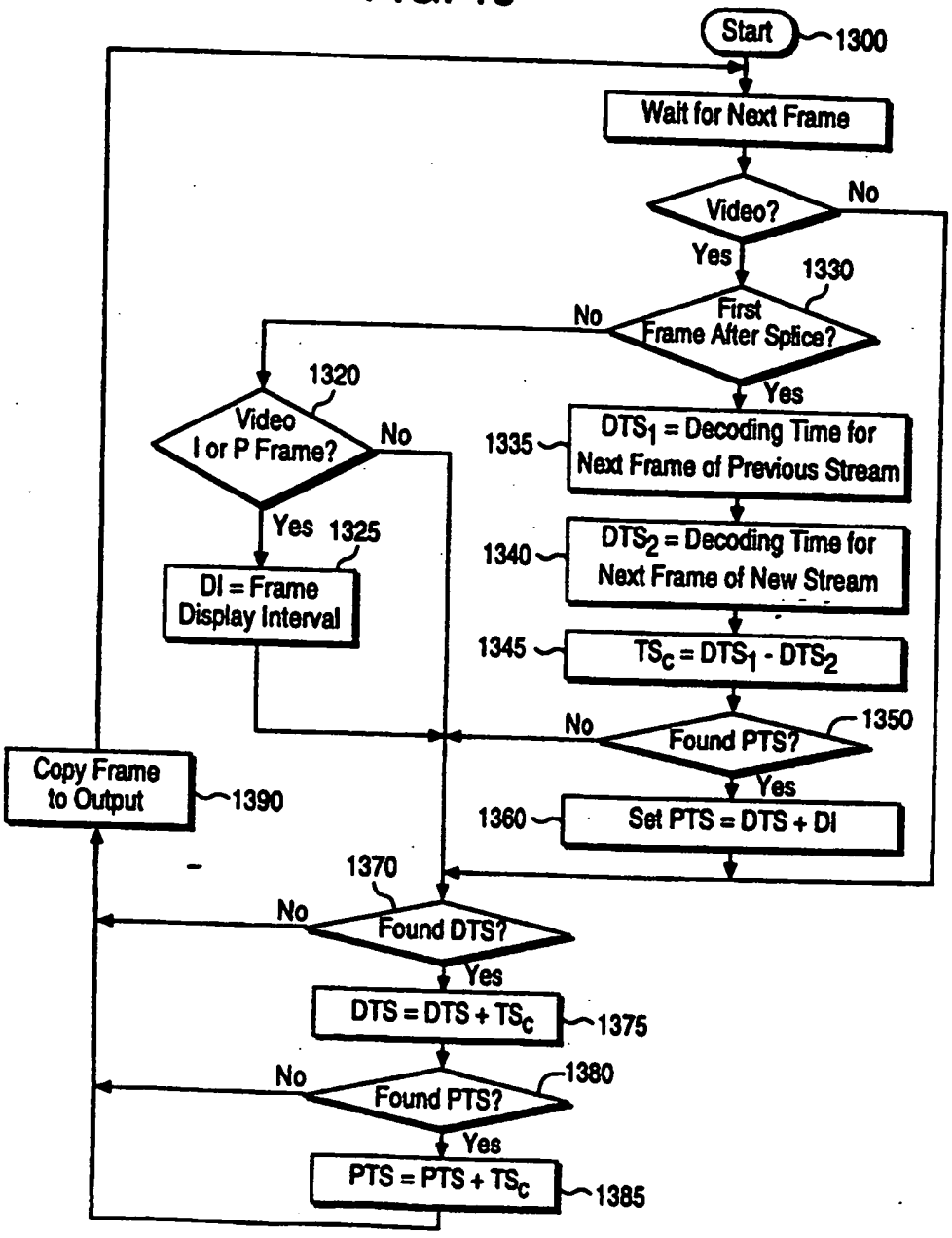

Start
1210
Extract Temporal Reference for Next Frame (I-Frame)
1220
TR_C = Temporal Reference
1230
Change Temporal Reference in Bit Stream to 0
1240
GOP Header?
1250
No
Yes
Set Closed GOP
1260
Wait for Next Frame
1270
GOP Header?
1280
No
Yes
End
Subtract TR_C From Temporal Reference in Bit Stream
1290

FIG. 12

FIG. 13


Start
1300
Wait for Next Frame
Video?
No
Yes
1330
First Frame After Splice?
No
Yes
1320
Video I or P Frame?
No
Yes
1325
DI = Frame Display Interval
1335
DTS1 = Decoding Time for Next Frame of Previous Stream
1340
DTS2 = Decoding Time for Next Frame of New Stream
1345
TSc = DTS1 - DTS2
No
Found PTS?
1350
Yes
1360
Set PTS = DTS + DI
Copy Frame to Output
1390
1370
No
Found DTS?
Yes
DTS = DTS + TSc
1375
No
Found PTS?
1380
Yes
PTS = PTS + TSc
1385

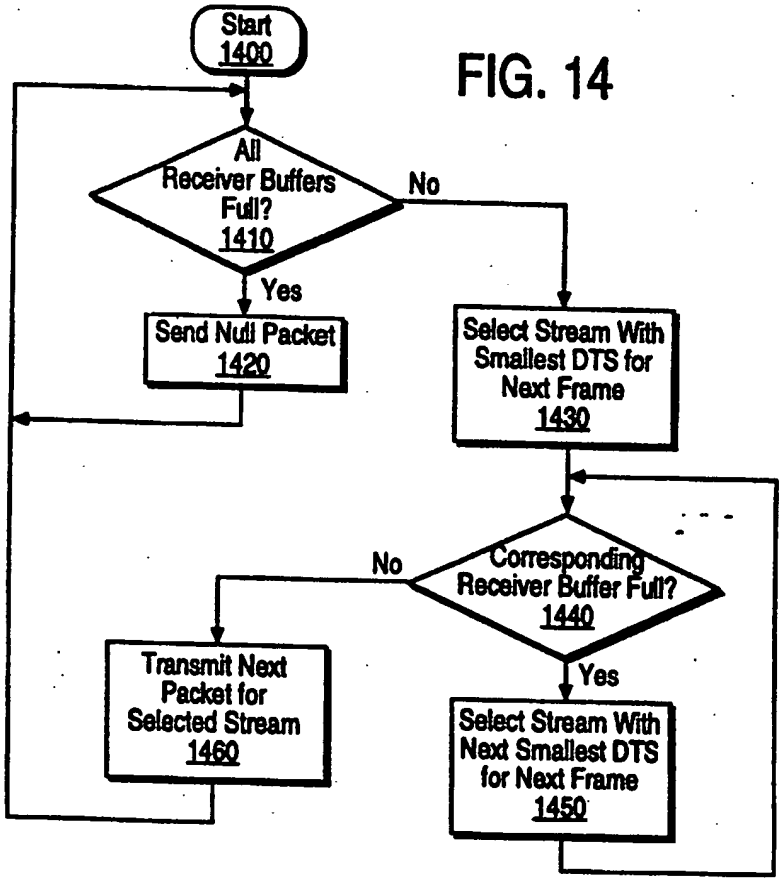

Start
1400
All Receiver Buffers Full?
1410
No
Yes
Send Null Packet
1420
Select Stream With Smallest DTS for Next Frame
1430
Corresponding Receiver Buffer Full?
1440
No
Yes
Transmit Next Packet for Selected Stream
1460
Select Stream With Next Smallest DTS for Next Frame
1450

FIG. 14